Higher-Order Demand-Driven Program Analysis (Artifact)

Leandro Facchinetti[*1], **Zachary Palmer**[2], **and Scott F. Smith**[3]

1 **The Johns Hopkins University**
Baltimore, MD, USA
leandro@jhu.edu
2 **Swarthmore College**
Swarthmore, PA, USA
zachary.palmer@swarthmore.edu
3 **The Johns Hopkins University**
Baltimore, MD, USA
scott@cs.jhu.edu

Abstract

This artifact is a proof-of-concept implementation of DDPA, an on-demand program analysis for higher-order functional programs. The implementation, written in OCaml, includes a parser, evaluator, and DDPA analysis for the language defined in the companion paper (including the proper record semantics extension). The analysis may be performed using different levels of precision as specified by the user and is capable of rendering the control flow graphs and pushdown systems using the GraphViz language DOT. This artifact was used to verify the conclusions of the companion paper and produces visualizations matching those figures in the companion paper's overview.

1998 ACM Subject Classification F.3.2. Semantics of Programming Languages

Keywords and phrases program analysis, polynomial, demand-driven, flow-sensitive, context-sensitive

Digital Object Identifier 10.4230/DARTS.2.1.9

Related Article Zachary E. Palmer and Scott F. Smith, "Higher-Order Demand-Driven Program Analysis", in Proceedings of the 30th European Conference on Object-Oriented Programming (ECOOP 2016), LIPIcs, Vol. 56, pp. 19:1–19:25, 2016.
http://dx.doi.org/10.4230/LIPIcs.ECOOP.2016.19

Related Conference 30th European Conference on Object-Oriented Programming (ECOOP 2016), July 18–22, 2016, Rome, Italy

1 Scope

This artifact serves as a proof-of-concept for the examples and formalism given in the companion paper. The artifact produces the same control flow graphs and outputs as discussed in the examples provided in the overview. It also implements the semantics of the proper record extension. This artifact does *not* implement the semantics of the state or deep pattern extensions.

This artifact is designed for transparency and correctness and not for performance. Its implementation is quite naïve (as discussed in the paper) and is particularly slow on larger programs.

* This co-author's work is supported by a scholarship provided by CAPES – Process number: 13477/13-7.

2 Content

The artifact package includes:

- OCaml source code for the proof-of-concept implementation of the DDPA algorithm,
- a collection of unit tests which demonstrate DDPA on examples from the paper and other sources,
- pre-generated results so that results matching the overview examples from the paper can be inspected without building the artifact, and
- detailed instructions for producing the artifact in a variety of ways, provided both as an `index.html` file as well as a `README.md` file.

Our build instructions explain how to build and run the proof-of-concept using OPAM (the OCaml package manager). As an alternative, these same files explain how to build the artifact using a preconfigured Docker image or by setting up a virtual machine with VirtualBox and Vagrant.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The latest version of our code is available on GitHub at `https://github.com/JHU-PL-Lab/odefa-proof-of-concept`.

4 Tested platforms

This artifact is known to work on the following platforms:

- Debian 8 (Jessie stable) using OPAM 1.2.0 and OCaml 4.02.2.
- Debian 9 (Stretch testing) using OPAM 1.2.2 and OCaml 4.02.2.
- OS X 10.11.3 (El Capitan) using OPAM 1.2.2 and OCaml 4.02.3.
- Any system capable of running Oracle VirtualBox 5.0.4 and Vagrant 1.7.3.

5 License

Apache 2.0 (`http://www.apache.org/licenses/LICENSE-2.0`)

6 MD5 sum of the artifact

88e98fe9019cb2bb1e758be0ca6325f9

7 Size of the artifact

8.6 MB